CSFinder: A Cold-Start Friend Finder in Large-Scale Social Networks

CSFinder: A Cold-Start Friend Finder in Large-Scale Social Networks

Yasser Salem, Jun Hong, Weiru Liu
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
Email: {ysalem01, j.hong, w.liu}@qub.ac.uk

***Abstract*—Recommending users for a new social network user to follow is a topic of interest at present. The existing approaches rely on using various types of information about the new user to determine recommended users who have similar interests to the new user. However, this presents a problem when a new user joins a social network, who is yet to have any interaction on the social network. In this paper we present a particular type of conversational recommendation approach, critiquing-based recommendation, to solve the cold start problem. We present a critiquing-based recommendation system, called CSFinder, to recommend users for a new user to follow. A traditional critiquing-based recommendation system allows a user to critique a feature of a recommended item at a time and gradually leads the user to the target recommendation. However this may require a lengthy recommendation session. CSFinder aims to reduce the session length by taking a case-based reasoning approach. It selects relevant recommendation sessions of past users that match the recommendation session of the current user to short-cut the current recommendation session. It selects relevant recommendation sessions from a case base that contains the successful recommendation sessions of past users. A past recommendation session can be selected if it contains recommended items and critiques that sufficiently overlap with the ones in the current session. Our experimental results show that CSFinder has significantly shorter sessions than the ones of an Incremental Critiquing system, which is a baseline critiquing-based recommendation system.**

***Keywords*-Web 2.0; Twitter; Social Networks; Conversational Recommendation; Critiquing; Recommender Systems.**

I. Introduction

Recommender systems help users select suitable items from a large collection of items with a range of features. Many recommendation techniques have been proposed, from *collaborative filtering* [1], [2], which uses simple ratings-based user profiles to generate recommendations from similar users, to *content-based* techniques [3], [4], which use the detailed knowledge of the recommended items to make recommendations.

Most recommender systems currently deployed use the *single-shot* strategy. They produce a ranked list of recommendations for the user [1], [2], [5], [6]. The single-shot strategy is suitable for the recommendation of simple products and services, such as ringtones, movies, books, etc., but it is not well suited for recommending items with complex features. In such cases, it is more effective to offer the user an opportunity to provide feedback, and to refine their needs and preferences. This has motivated researchers to develop so called conversational recommendation approaches [7], [8]. One of these approaches is called critiquing-based recommendation, in which the user can provide their feedback by critiquing the features of recommended items. The user participates in an iterative recommendation conversation, in each round of which the user receives recommendations and critiques a feature of the recommended items. This process continues with another round of conversation until the user has reached the target recommendation.

Critiquing-based recommendation has proven to be an effective approach to conversational recommendation. However they have a tendency to produce protracted recommendation sessions, due to the limited feedback that critiques can provide. Recent work has concentrated on reducing the session lengths of critiquing-based recommender systems [9], [10], [11], [12], [13].

Social networking platforms such as Twitter have huge numbers of users. Twitter has 284 millions of active users each month[1]. A report from Twopcharts[2] in April 2014 states that about 44% of the 974 millions of the existing Twitter accounts have never sent a tweet. The report further states that 30% of the existing Twitter accounts have sent only 1-10 tweets. Twitter users are usually following rather than being followed.

Finding interesting users to follow among millions of Twitter users is a difficult task, which may be part of the reason for the inactivity of some Twitter users [14], [15], [16]. Various approaches have been proposed for the task [17], [18], [19], [20], [21], [22]. These approaches typically rely on analysing the followees, followers and tweets of a user in order to recommend other users for the user to follow, who have interests relevant to the user. However, when a new user joins Twitter, the user has no tweets, followees and followers. In this paper we present a particular type of conversational recommendation approach, critiquing-based recommendation, to solve the cold start problem.

We take an experience-based approach, called CSFinder, to recommend users for a new user to follow. A traditional critiquing-based recommendation system allows a user

[1]twitter.com, Jan. 2015
[2]twopcharts.com

to critique a feature of a recommended item at a time and gradually leads the user to the target recommendation. However this may require a lengthy recommendation session. CSFinder aims to reduce the session length by taking a case-based reasoning approach. It selects relevant recommendation sessions of past users that match the recommendation session of the current user to shortcut the current recommendation session. It selects relevant recommendation sessions from a case base that contains the successful recommendation sessions of past users. A past recommendation session can be selected if it contains recommended items and critiques that sufficiently overlap with the ones in the current session. Our experimental results show that CSFinder has significantly shorter sessions than the ones of an Incremental Critiquing system, which is a baseline critiquing-based recommendation system.

The rest of the paper is organised as follows. Section 2 describes related work on different types of recommender systems and techniques in general and Twitter user recommendation in particular. Sections 3 and 4 present our new approach to the cold-start friend finder problem on Twitter, which harnesses the critiquing experiences of past users to generate new recommendations and shorten recommendation sessions. Section 5 presents the results of our experimental evaluation to demonstrate the performance and potential benefits of our new approach.

II. Related Work

Conversational recommender systems have adopted various forms of feedback, including value elicitation, ratings-based feedback, preference-based feedback and critiquing [23]. Systems which employ value elicitation ask users to enter specific values of product features, e.g., *hard-drive = 320GB*. Though this is a rich form of feedback, the user must possess a high level of domain knowledge to use it effectively. In contrast, ratings-based feedback is a much simpler form of feedback, preferred by most collaborative filtering systems. Users assign a simple rating, e.g., *3 out of 5 stars*, to indicate the level of their satisfaction with the recommendation [5]. With ratings-based feedback, the user does not require detailed domain knowledge, since they are not commenting on specific features. Instead they simply provide an overall recommendation rating. Preference-based feedback is a special case of ratings-based feedback, in which, instead of rating a set of recommendations, the user simply indicates their preferred recommendation [24]. This is a low cost form of recommendation that requires little domain knowledge, just an ability to distinguish good recommendations from bad ones. However, it is clearly limited in its information content, as it is not always apparent why the user has selected one recommendation over others. In this paper, we look at an altogether different form of feedback, critiquing, which strikes a balance between ease of feedback and the information content of the feedback. Simply put, critiquing allows users to indicate a *directional preference* with respect to a particular feature of the recommendation. For example, in the scenario of a Twitter recommender system a user might respond to a given recommendation by asking for new recommended users whose tweets are on *Politics*. In this case the user is critiquing the *Topics* feature, asking for new users who tweet in politics. This is the standard form of critiquing, which is also called *unit critiquing* because the user critiques a single feature at a time. The critique acts as a filter over the remaining users to be recommended. The next recommendation will be compatible with the current critique while being maximally similar to the current recommendation.

A previous approach to critiquing, *Incremental Critiquing (IC)* [25], maintains a user model made up of the user's previous critiques in a recommendation session. The user model is then used to guide the selection of new recommendations in the session. For example, suppose a camera shopper has already provided a number of critiques. The user model may indicate that the shopper is looking for a *DSLR* camera that is *cheaper than $500*, *manufactured by Nikon*. [25] describes how a user model is generated from such a sequence of critiques. When the user provides a new critique, perhaps looking for a *resolution higher than 5 mega-pixels*, instead of just selecting a new recommendation based on its similarity to the current recommendation and its compatibility with its current critique, the user model is further used to rank all candidate recommendations. The end result is that the new recommendation is guaranteed to maximally match the user's previous critiques (via the user model). It has been shown that this approach produces more focused recommendations that lead to significantly shorter recommendation sessions.

There are many approaches to recommending Twitter users, such as Twittomender [17], [18], which is a web service in which the user must sync their Twitter account with Twittomender, so that the user can be modelled by their last 200 tweets, the tweets of their followees, and the tweets of their followers. Twittomender can then generate a profile of the user.

Twittomender uses a combination of content-based and collaborative filtering approaches to recommendation. In terms of content-based techniques, users are represented by the tweets of their own, followees and followers. In terms of collaborative filtering, users are represented by the IDs of their followees and followers. It is shown that the collaborative filtering approach is better at finding relevant followees than the content-based approach.

Armentano et al. [19], [20], [21] proposed an approach that differs slightly from Twittomender in that it does not require the profiles of Twitter users. It explores the follower/followee network in order to find users to recommend. They compared their own approach to Twittomender and showed better general precision than Twittomender. Their

Features			
User name	Faisal Al-Kassim	The New York Times	BBC Breaking News
Location	☒ Qatar ☒	☒ New York ☒	☒ London ☒
Languages	☒ Arabic ☒	☒ English ☒	☒ English ☒
Topics	☒ Politics ☒	☒ News ☒	☒ News ☒
Tweets	↓ 48.6 K ↑	↓ 164 K ↑	↓ 23 K ↑
Avg Tweets daily	↓ 54 ↑	↓ 54 ↑	↓ 8 ↑
Tweet rank	↓ 1,325 ↑	↓ 49 ↑	↓ 65 ↑
Followers #	↓ 2.08 M ↑	↓ 14.8 M ↑	↓ 13.1 M ↑
Following #	↓ 1,447 ↑	↓ 963 ↑	↓ 3 ↑
joined Twitter	↓ 1,862 days ↑	↓ 2,877 days ↑	↓ 2,862 days ↑
Avg daily tweets last month	↓ 116 ↑	↓ 116 ↑	↓ 23 ↑
New followers last month	↓ 83,078 ↑	↓ 437,813 ↑	↓ 454,261 ↑
	Follow	Follow	Follow

Figure 1: Interface of a Critiquing-based Recommendation System

approach consists of two main steps: 1. Finding a suitable group of candidate users: If user X follows user Y, and user Z also follows user Y, then the system considers those followees of Z, whom X is yet to follow as a group of candidate users for recommendation to X. 2. The tweets of each candidate user for recommendation to X are analysed to determine if they are of interest to X. This can be determined by the similarity between the tweets of X and the ones of the candidate.

The CB-MF method proposed in [22] uses both follower and followee relationships to discover communities to improve user recommendation. The CB-MF method uses a two-phase approach in which user communities are first discovered based on the topics of their tweets, and then a matrix is applied to each community to generate a list of candidate followees.

The main limitation of Twittomender, Armentano et al's approach and the CB-MF approach is that they cannot generate a user profile for a new user or an existing user who doesn't have much interaction on Twitter in terms of tweets, followees and followers [26], [27], [28], [29]. This leads to the 'Cold-Start' problem which is a common problem in recommendation systems: there is relatively little information about a new user. The cold-start problem arises when a new entity enters the system for the first time. In this situation the recommendation system cannot make recommendations because there is little information about the new entity.

III. Cold-Start Friend Finder

In this section, we present an experience-based critiquing approach that improves on the standard critiquing-based recommendation in terms of session lengths. A standard critiquing-based recommendation session consists of a series of recommendation-critiquing interactions, in each of which the user receives a recommendation and decides whether they are satisfied with the recommendation; If not, the user is expected to respond by issuing a critique, e.g., *hard-drive* $> 320GB$. The user's critique helps the system to decide on what to be recommended in the next interaction. The system then makes another recommendation in the next interaction. This process continues until the user has reached their target recommendation or stops.

In the same spirit of case-based reasoning [30], [31], experience-based critiquing harnesses a new source of knowledge, namely the critiquing experiences of past users.

The basic assumption is that the successful recommendation sessions of past users must encode useful sequences of critiques, which may help to short-cut the recommendation session of the current user.

In the remainder of the paper, we describe the CSFinder technique and how it leverages the past experiences in a critiquing-based recommender system. We go on to demonstrate the potential of these experiences to significantly improve the efficiency on incremental critiquing (IC) [25].

Figure 1 shows the interface of a typical critiquing-based recommender system, which presents the current recommendation with a range of features. For each feature, the critiques are presented on both side of its value. The critique acts as a filter over the remaining recommendations.

An example run of our system for recommending a followee to a new user is as follows:
Step 1: The user selects a topic from the topic list, e.g., *topics = Dietitian*.
Step 2: The user wants a followee with another feature, e.g., *Followers > 5,000*, perhaps to ensure that the recommended user is popular.
Step 3: The user wants a followee with another feature, e.g., *When joined Twitter > a year*, to ensure that the recommended user is well established.
Step 4: The user wants a followee with another feature, e.g., *Number of tweets > 7000*, to ensure that the recommended user is active.
Step 5: The user wants a followee with another feature, e.g., *Avg Tweets last month > 100*, to ensure that the recommended user has been active recently.
Step 6: The user wants a followee with another feature, e.g., *location = UK*, to ensure that the recommended user understands the people's health issues in that particular location.
Step 7: The user wants a followee with another feature, e.g., *Languages = English*.
Step 8: The user wants a followee with another feature, e.g., *New followers last month > 1,000*, to ensure that the recommended user is gaining more followers recently.

Critiquing-based recommender systems aim to provide the user with a set of easy-to-understand critiques. They benefit the user since shorter sessions mean less effort for the user and improved conversion rate for the recommender system.

A. Recommendation Sessions

In a typical critiquing-based recommendation session the user will start with a high-level understanding of their needs. For example, when finding a Twitter user to follow they might start by indicating a language and a location. During the course of the session their needs will be refined, as the user critiques the features of recommended Twitter users, perhaps indicating that they are looking for someone who is not in the UK but has more followers than earlier recommendations. Thus, during the session the user may provide feedback on a range of different features.

We can model each recommendation session, s_i, as a sequence of *recommendation-critique pairs*, as shown in Fig. 2 and Equations 1-2, where r_i represents a recommendation and c_i represents the critique that is applied by the user to the recommendation. Each c_i is represented as a triple, $(f_i, v_i, type_i)$, as shown in Equation 4, where f_i is one of the features for r_i, i.e., $f_i \epsilon r_i$, which the critique applies to, v_i is a value that the critique applies and $type_i$ is the type of the critique (typically, $type_i \in \{<, >, =, <>\}$). We assume that each session terminates (as shown in Equation 3) when the user chooses to *accept* a recommendation, indicating that they are satisfied with the recommendation, or when they choose to *stop* the session, presumably because they have grown frustrated with the recommendations received. Thus we can add *accept* and *stop* to the set of permissible critique types such that every session terminates with one of these types.

$$s_i =< p_1, ..., p_n > \tag{1}$$

$$p_i = (r_i, c_i) \tag{2}$$

$$terminal(s_i) = p_n = (r_n, c_n), c_n \in \{accept, stop\} \tag{3}$$

$$c_i = (f_i, v_i, type_i) \tag{4}$$

In general, a critiquing-based recommender system has many users who will produce a large collection of critiquing sessions as they engage with the recommender system. The sessions reflect the *experiences* of these users and capture potentially useful knowledge about their preferences and the different trade-offs they tend to explore. In this paper, we are interested in the potential of these experiences to inform the recommendation process itself. In other words, these critiquing sessions are the cases in a case base of critiquing experiences.

For the remainder of this paper we will assume that only *successful sessions* — that is, those sessions where the user *accepts* a recommendation — are stored in the case base. We can treat the accepted recommendation as the *solution* of the case and the recommendation-critique pairs that proceed it as the *specification* of the case. We will describe how to harness these critiquing experiences to improve the efficiency of the recommendation process by using a new critique-based recommendation approach, called CSFinder, which differs from the traditional approach to critiquing in terms of how new recommendations are generated; see Fig. 3 for a brief overview.

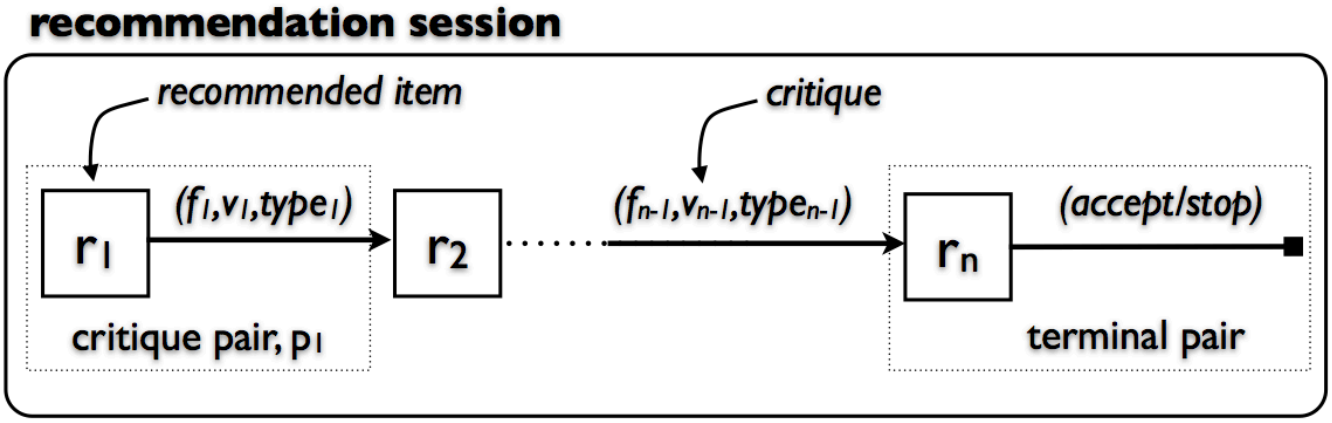

Figure 2: Each recommendation session is made up of a sequence of recommendation-critique pairs.

B. Conventional Critiquing

In a conventional critiquing-based recommender system, when a user applies a critique c_i to a recommended item r_i, the recommender responds by retrieving an item, r_T, which is compatible with c_i, in the sense that the item satisfies the critique c_i, and maximally similar to r_i, as in Equations 5-6. Note that $r.f$ represents the value of feature f in recommended item r. $apply(type, u, v)$ is true if and only if the predicate denoted by $type$ is satisfied by the values u and v; for example, $apply(<, 2500, 4000)$ is $true$ whereas $apply(=, English, Arabic)$ is not.

$$r_T = Recommend(r_i, c_i) = \underset{\forall\, r_j \epsilon\, items \,\wedge\, satisfies(c_i, r_j)}{argmax} \Big(sim(r_i, r_j) \Big) \tag{5}$$

$$satisfies(c_i, r_j) \leftrightarrow apply(type_i, r_j.f_i, r_i.f_i) \tag{6}$$

C. Harnessing CSFinder Recommendation

CSFinder extends conventional critiquing by reusing past sessions to guide the critiquing process. Instead of retrieving a new item that is maximally similar to the current recommendation, and compatible with the user's critique, we recommend one of the items that past users have ended up accepting in similar critiquing sessions.
This can be best understood in terms of three basic steps:
(1) Identifying past critiquing sessions that are similar (i.e., relevant) to the current session.
(2) Ranking recommendation candidates from the terminal items of the identified similar sessions.
(3) Filtering the ranked candidates to eliminate those that do not satisfy the current user's critiques.

IV. Identifying Relevant Critiquing Sessions in CSFinder

When a user applies a critique c_i to a recommended item r_i the user's current (partial) critiquing session, $(r_1, c_1), ..., (r_i, c_i)$, is used as a query ($q_T$) over the case base of past critiquing sessions, in order to identify a set of *relevant sessions*; see (a) and (b) in Fig. 3. Briefly, a relevant session is one that has at least some overlap with the current query (see Equation 8, where typically $t = 0$), based on a particular overlap metric. In this case, we propose an overlap metric, as shown in Equation 7, which computes an overlap score based on both the number of recommendation pairs (item and critique) and the number of critiques (critique only) in q_T, which are also present in a given session. The recommendation pairs (item and critique) in q_T that are present in the given session are given a weight of 2, while the critiquess (critique only) in q_T that are also present in the given session are given a weight of 1. Finally, we use Equation 8 to select relevant sessions. This process of selecting relevant sessions favours more recommendation pairs while still having the benefit of critique overlap to improve performance.

Note that in the case that there are no relevant sessions, and no candidates to recommend, the system reverts to incremental critiquing (IC) and retrieves a new item that is maximally similar to the current recommendation and compatible with the user's critiques.

$$OverlapScore(q_T, s_i) = (2 \times \sum_{(r_i, c_i) \in q_T} \sum_{(r_j, c_j) \in s_i} match((r_i, c_i), (r_j, c_j))) + (1 \times \sum_{c_i \in q_T} \sum_{c_j \in s_i} match(c_i, c_j)) \tag{7}$$

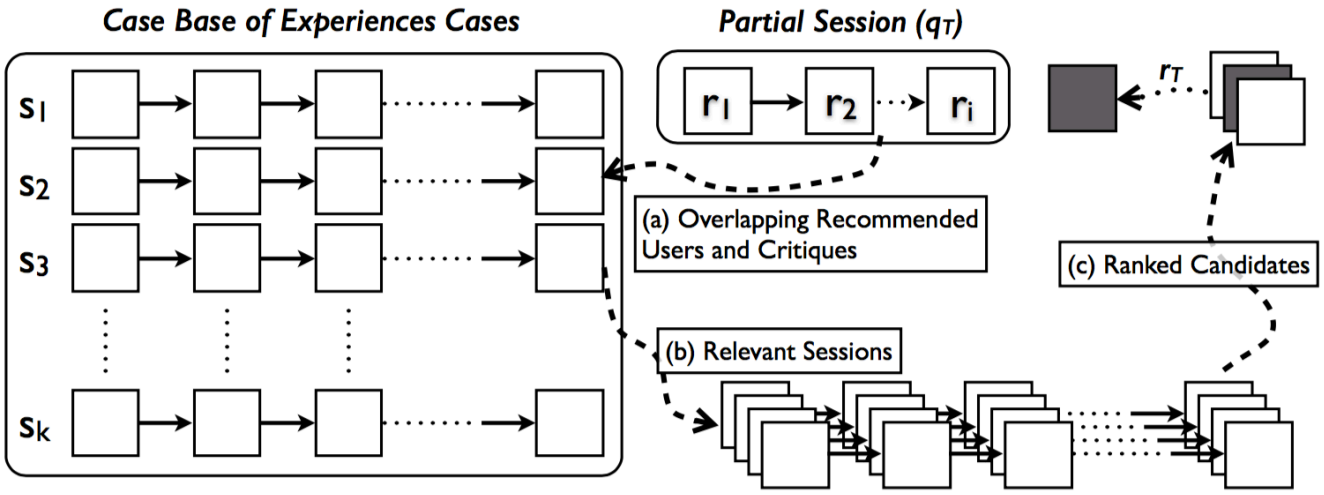

Figure 3: CSFinder selects those past recommendation sessions that contain recommended users and critiques sufficiently overlapping with the ones in the current recommendation session. Those recommended Twitter users who have been accepted in these relevant past sessions are identified as the candidates of the recommended Twitter user. The identified recommendation candidates are ranked and filtered. The top-ranked remaining Twitter user is recommended to the user.

$$S^{REL} = RelevantSessions(q_T, S) = \left\{ s_i \epsilon S : OverlapScore(q_T, s_i) > t \right\} \quad (8)$$

$$RecScore(r_F, q_T, S^{REL}) = \sum_{\{\forall s_i \epsilon S^{REL} : r_F = terminal(s_i)\}} OverlapScore(q_T, s_i) \quad (9)$$

A. Ranking Recommendation Candidates

The identified relevant sessions (S^{REL}) have previously led a past user to a successful recommendation. Each relevant session terminates with an accepted recommendation r_F, which forms a candidate for the next recommendation in the current session. Generally speaking, an accepted recommendation may be associated with more than one past session. Intuitively, it makes sense to give preference to a recommended item that is accepted more often in relevant sessions. Therefore, the recommendation candidates can be ranked based on their scores as defined in Equation 9; see (c) in Figure 3.

B. Filtering Conflicting Candidates

There is no guarantee that the ranked recommendation candidates are compatible with the user's critiques. Thus, in the final step incompatible recommendation candidates are eliminated. The simplest way to do this is to eliminate those candidates that fail to satisfy *all* of the user's critiques so far, $(r_1, c_1), ...(r_i, c_i)$, in the current session. However, this is not ideal since in many cases, users may change their mind during a session, resulting in conflicting critiques in the session [25]. For example, a user might start by looking for a product that is *cheaper than $100* only to later shift towards looking for a product in the $100 - $150 range. In this case, eliminating recommendation candidates in the $100 - $150 range, based on the earlier critique in the current session, would be inappropriate.

Accordingly we edit the critiques in the current session by working backwards through the session starting with the most recent critique. If a critique is in conflict with a more recent critique (that has already been processed) then it is eliminated. This leaves a set of core critiques which represent the *boundaries* of the current user's preferences with respect to the features that have already been critiqued. Items that do not satisfy the core critiques are eliminated from the ranked list of recommendation candidates and the top-ranked remaining candidate, shown as r_T in Figure 3, is recommended to the user.

V. Experimental Evaluation

To evaluate the CSFinder approach we have developed a Twitter user recommender system and evaluated it, based on a comprehensive database of Twitter users.

A. Datasets

For the purposes of the evaluation, two datasets are used: *Twitter users* as recommended items and the successful recommendation sessions of past users as *experience cases*. We crawled Twitter user information in *twopcharts.com* to create the Twitter users dataset consisting of 650,000 users. We filtered the Twitter users dataset to retain those

active users only, who tweeted on average at least twice daily. We also removed those Twitter users who did not gain more followers over the past *3* months. The final Twitter users dataset contains a total of 124,545 Twitter users. Each user is represented by 17 different features (e.g., Location, Language, Followers, Followings, when joined twitter, number of tweets, average tweets daily, number of new followers last month, number of new followings last month, number of average tweets daily last month, etc.), including 4 nominal features and 13 numeric features.

We divided the 125,000 Twitter users into three groups of *10,000, 50,000 and 125,000 users* to evaluate CSFinder's performance with different sizes of datasets.

Ideally, we would like to evaluate CSFinder using real users. However, this is not currently feasible since it would require a major live deployment over an extended period of time. Instead we adopted the approach taken by [10], [12], [32], [33] to automatically generate critiquing sessions by simulating the behaviour of a rational user, using incremental critiquing (IC) [25]. For each user in each of the three different user datasets of 10,000, 50,000 and 125,000 users, we ran IC with it as the target user, simulating the critiques of a rational user, until the target user was recommended by IC, at which point we recorded the recommendation session. We randomly selected 2-3 features from each target user, which acted as the starting point of a recommendation session. The artificial user then critiqued a feature at a time. To do this it automatically selected one of the features of the recommended user and critiqued it *in the direction* of the target user. For example, if the target user has 7,500 followers and the recommended user has 4,000 followers, then if the followers feature is selected for critiquing, the *more followers* ($>$) critique will be applied on the followers feature. We can generate an arbitrary number of recommendation sessions. In our experiments, however, for each target user, we generated 1, 3, 5 and 7 recommendation sessions respectively to create the case bases of 1, 3, 5, and 7 times of the size of the appropriate user dataset. The case bases of different sizes were then used for evaluating CSFinder.

B. Algorithms & Methodology

We compared the performance of our experience-based critiquing recommendation with incremental critiquing (IC) recommendation [25], which is a baseline critiquing-based conversational recommendation system. We generated a separate set of 500 target users as our *test set*. For each of the target users we ran CSFinder, simulating the critiques of a rational user, until the target user was recommended by CSFinder. We automatically created a *query*, by selecting 2 features from the target user at random, which acted as the starting point of a recommendation session. The artificial user then critiqued a feature at a time. To do this it automatically selected one of the features of the recommended Twitter user and critiqued it *in the direction* of the target Twitter user. Moreover, features were selected for critiquing based on a probability model that favours nominal and numeric features over binary features to simulate a more realistic critiquing session. The probability model gives each of the nominal and numeric features twice of the chance over binary features. A target user is deemed to be found once the target user has been recommended, at which point we noted the session length.

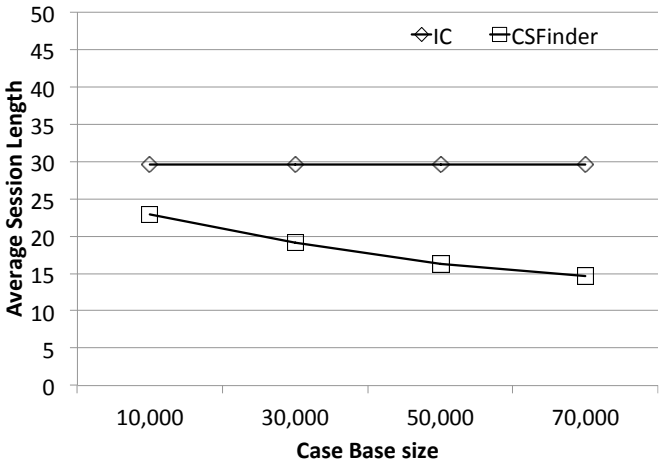

Figure 4: The average session lengths of CSFinder and IC with a single recommendation per cycle and a dataset of 10,000 Twitter users.

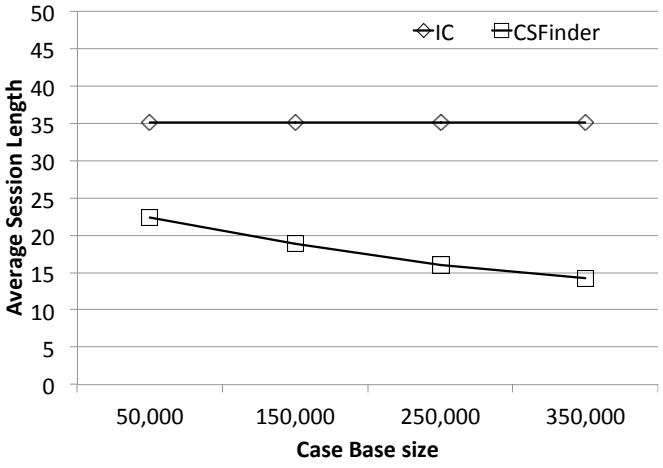

Figure 5: The average session lengths of CSFinder and IC with a single recommendation per cycle and a dataset of 50,000 Twitter users.

C. Results

The key efficiency issue to consider is whether CSFinder leads to earlier recommendations and thus short-cut recom-

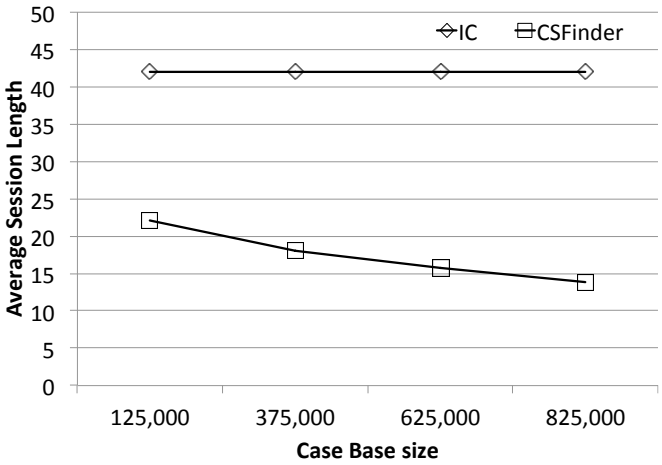

Figure 6: The average session lengths of CSFinder and IC with a single recommendation per cycle and a dataset of 125,000 Twitter users.

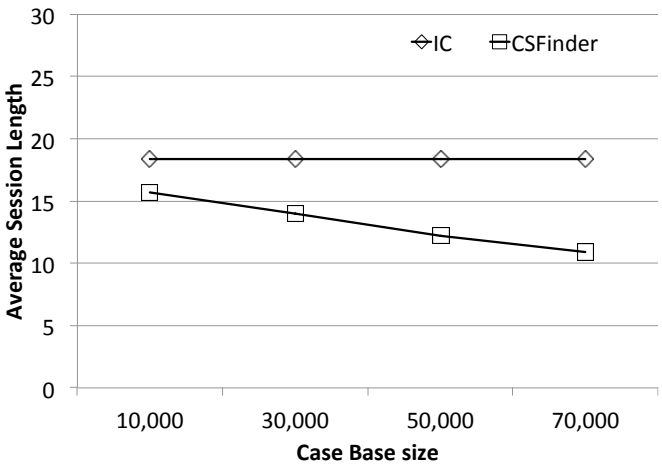

Figure 7: The average session lengths of CSFinder and IC with three recommendations per cycle and a dataset of 10,000 Twitter users.

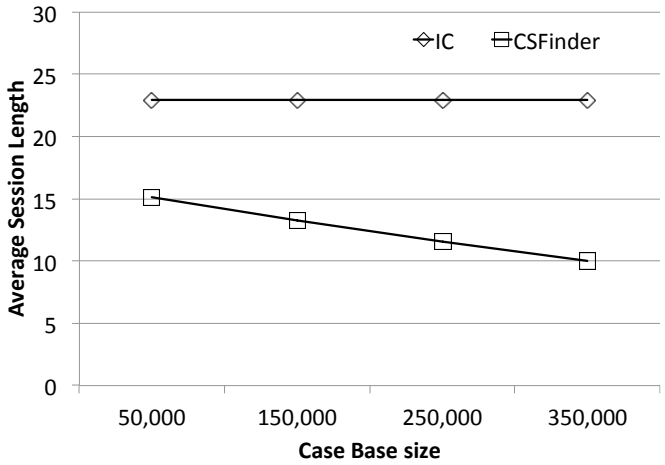

Figure 8: The average session lengths of CSFinder and IC with three recommendations per cycle and a dataset of 50,000 Twitter users.

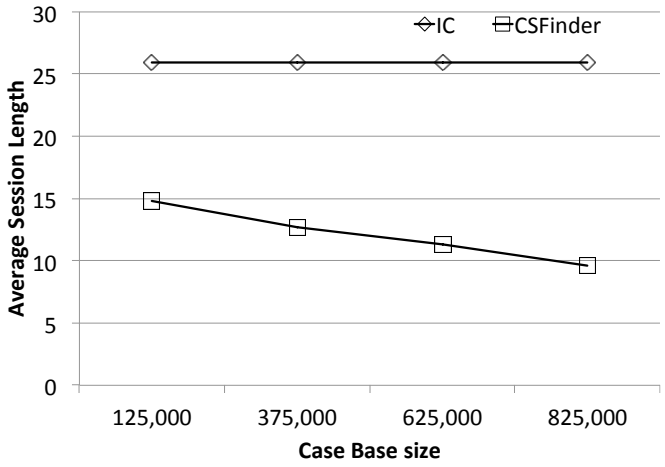

Figure 9: The average session lengths of CSFinder and IC with three recommendations per cycle and a dataset of 125,000 Twitter users.

mendation sessions, when compared to IC. If so CSFinder can lead to some tangible benefits for the user. All the other things being equal, shorter recommendation sessions mean less effort for the user and improved conversion rate for the recommender system. Figure 4 shows the average session lengths of both CSFinder and IC when a dataset of 10,000 users was used and a single recommendation was made in each recommendation-critique round. As shown, CSFinder achieves much more reduction in session length: for the largest case base of 70,000 recommendation sessions, the average session length of CSFinder is reduced to under 15 cycles (compared to 30 for IC).

When comparisons were made between IC and CSFinder on larger datasets of Twitter users, the benefits of CSFinder over IC were even greater, as shown in Figure 5 and 6. CSFinder has much more benefits over IC across various sizes of user datasets, ranging from 48% for the dataset of 50,000 users to 67% for the dataset of 125,000 users.

We have so far considered one recommendation in each recommendation cycle. However, the idea that the presentation of multiple recommendations in each cycle would give the user a selection of recommendations to choose

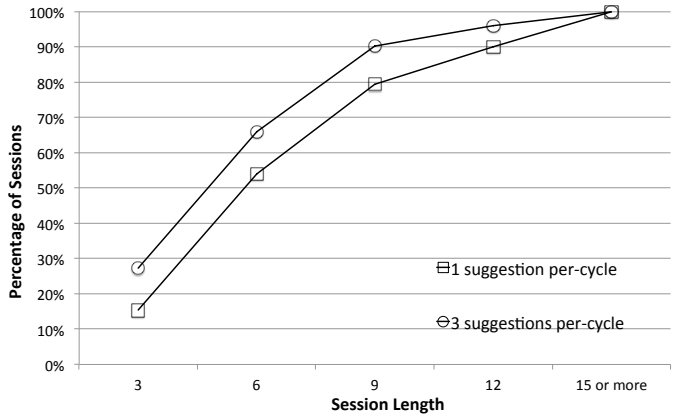

Figure 10: The percentage of sessions that reached the target.

from merits further investigation. Allowing the user to view multiple recommendations in each cycle may help them understand the experience cases in the case base and at the same time help them home in on their final recommendation. We are aware that in a real-world system there may be issues on the presentation of multiple recommendations at once. However an artificial simulation provides the perfect opportunity to test the idea (the artificial user chooses recommendations in line with their target). As a baseline case for comparison, Figures 7, 8 and 9 show average session lengths when three recommendations are made in each recommendation cycle. As shown in Figures 7, 8 and 9, both CSFinder and IC gain reduction in session lengths when multiple recommendations are presented in each cycle, with CSFinder having session reduction to just under 10 cycles with the full dataset of 125,000 users. These preliminary results show that the idea of preference-based critiquing is worth being further investigated in future research.

In addition to average session reduction, Figure 10 shows the percentage of target users who have been successfully reached as a recommendation after a certain number of recommendation cycles. For example, more than 50% of the target users have been reached in just 6 or less cycles, when the full dataset of 125,000 users and the largest case base of 825,000 recommendation sessions are used.

VI. Conclusions

In this paper we have presented CSFinder, a new critiquing-based approach to conversational Twitter user recommendation. Critiquing-based approach strikes a balance between ease of feedback and the information content of the feedback. It allows users to indicate a directional preference with respect to a particular feature of the recommendation. CSFinder supports more efficient critiquing by matching the critiquing session of the current user to the relevant successful recommendation sessions of past users and selecting new recommendations from the accepted recommendations in the identified relevant past recommendation sessions.

The results of our experiments are promising, showing a marked improvement over IC. These improvements are positively correlated with both the complexity of the user dataset and the size of the case base.

References

[1] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*, 2011, pp. 107–144.

[2] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[3] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, 2011, pp. 73–105.

[4] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[5] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying Collaborative Filtering to Usenet News." *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[6] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW 94)*. North Carolina, USA: ACM Press, August 1994, pp. 175–186.

[7] R. Burke, K. Hammond, and B. Young, "The FindMe Approach to Assisted Browsing," *Journal of IEEE Expert*, vol. 12(4), pp. 32–40, 1997.

[8] L. Chen and P. Pu, "Critiquing-based recommenders: survey and emerging trends," *User Model. User-Adapt. Interact*, vol. 22(1-2), pp. 125–150, 2012.

[9] M. Mandl and A. Felfernig, "Improving the performance of unit critiquing." in *UMAP*, ser. Lecture Notes in Computer Science, J. Masthoff, B. Mobasher, M. C. Desmarais, and R. Nkambou, Eds., vol. 7379. Springer, 2012, pp. 176–187.

[10] K. McCarthy, Y. Salem, and B. Smyth, "Experience-Based Critiquing: Reusing Critiquing Experiences to Improve Conversational Recommendation," in *Proceedings of the 18th International Conference on Case-Based Reasoning, (ICCBR-2010)*, ser. Lecture Notes in Computer Science, vol. 6176. Springer, 2010, pp. 480–494, alessandria, Italy.

[11] Y. Salem and J. Hong, "History-Aware Critiquing-Based Conversational Recommendation," in *Proceedings of the 22nd international conference companion on World Wide Web WWW2013*. ACM, 2013, pp. 63–64, rio de Janeiro, Brazil.

[12] Y. Salem, J. Hong, and W. Liu, "History-guided conversational recommendation," in *Proceedings of the 23rd international conference on World Wide Web companion (WWW '14)*. International World Wide Web Conferences Steering Committee / ACM, 2014, pp. 999–1004.

[13] H. Xie, L. Chen, and F. Wang, "Collaborative compound critiquing," in *UMAP*, 2014, pp. 254–265.

[14] Y. S. Kim, A. Mahidadia, P. Compton, A. Krzywicki, W. Wobcke, X. Cai, and M. Bain, "People-to-people recommendation using multiple compatible subgroups," in *Australasian Conference on Artificial Intelligence*, 2012, pp. 61–72.

[15] M. Rowe, M. Stankovic, and H. Alani, "Who will follow whom? exploiting semantics for link prediction in attention-information networks," in *International Semantic Web Conference (1)*, vol. 7649. Springer, 2012, pp. 476–491.

[16] J. Subercaze, C. Gravier, and F. Laforest, "Towards an expressive and scalable twitter's users profiles," in *Web Intelligence*, 2013, pp. 101–108.

[17] J. Hannon, M. Bennett, and B. Smyth, "Recommending twitter users to follow using content and collaborative filtering approaches," in *RecSys*, 2010, pp. 199–206.

[18] J. Hannon, K. McCarthy, and B. Smyth, "Finding useful users on twitter: Twittomender the followee recommender," in *ECIR*, 2011, pp. 784–787.

[19] M. Armentano, D. Godoy, and A. Amandi, "A topology-based approach for followees recommendation in Twitter," in *Proceedings of the Intelligent Techniques for Web Personalization and Recommendation (ITWP'11) at the 22nd International Joint Conferences on Artificial Intelligence (IJCAI 2011)*, pp. 22–29.

[20] ——, "Towards a followee recommender system for information seeking users in Twitter," in *Proceedings of the International Workshop on Semantic Adaptive Social Web (SASWeb'11) at the 19th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2011)*, 2011.

[21] M. G. Armentano, D. Godoy, and A. A. Amandi, "Followee recommendation based on text analysis of micro-blogging activity," *Inf. Syst.*, vol. 38, no. 8, pp. 1116–1127, 2013.

[22] G. Zhao, M. L. Lee, W. Hsu, W. Chen, and H. Hu, "Community-based user recommendation in uni-directional social networks," in *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management (CIKM '13)*. New York, USA: ACM, 2013, pp. 189–198.

[23] B. Smyth and L. McGinty, "An Analysis of Feedback Strategies in Conversational Recommender Systems." in *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Cognitive Science (AICS-2003)*, P. Cunningham, Ed., 2003.

[24] L. McGinty and B. Smyth, "Comparison-Based Recommendation," in *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR 2002)*, S. Craw, Ed. Springer, 2002, pp. 575–589.

[25] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth, "Incremental critiquing," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 143–151, 2005.

[26] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu, "Collaborative personalized tweet recommendation," in *SIGIR*, 2012, pp. 661–670.

[27] X. Ding, X. Jin, Y. Li, and L. Li, "Celebrity recommendation with collaborative social topic regression," in *IJCAI*, 2013.

[28] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: the who to follow service at twitter," in *WWW*, 2013, pp. 505–514.

[29] Y. S. Kim, A. Krzywicki, W. Wobcke, A. Mahidadia, P. Compton, X. Cai, and M. Bain, "Hybrid techniques to address cold start problems for people to people recommendation in social networks," in *PRICAI*, 2012, pp. 206–217.

[30] J. Kolodner, *Case Based Reasoning*. Morgan Kaufman Publishers, 1993.

[31] D. B. Leake, "Case-based reasoning," in *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., pp. 196–197.

[32] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth, "On the dynamic generation of compound critiques in conversational recommender systems," in *Proceedings of the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH-2004)*, ser. Lecture Notes in Computer Science, P. D. Bra and W. Nejdl, Eds., vol. 3137. Springer, 2004, pp. 176–184.

[33] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth, "Dynamic critiquing," in *Proceedings of the Seventh European Conference on Case-Based Reasoning (ECCBR-04)*, ser. Lecture Notes in Computer Science, P. Funk and P. González-Calero, Eds., vol. 3155. Springer, 2004, pp. 763–777.